DECENTRALIZED SOFTWARE UPGRADING IN REMOTE NETWORK BASED ON RASPBERRY PI

M. Srikanth **M .Vijaya Lakshmi**

1. **M.Srikanth, Department of ECE, St.Martin's Engineering College, Telangana, India, Email: srikanthmanasani@gmail.com**

2. **M. Vijaya Lakshmi, M.Tech., Assistant Professor, St.Martin's Engineering College, Telangana, India, Email: mvijayalakshmi78@gmail.com**

Abstract: With the development of virtualization technology, thin clients used in desktop cloud are taking place of personal computer (PC). Traditional PCs widely distributed call for huge maintenance for large enterprises, such as telecom operators, banks and so on. PCs face the challenge of modern IT system, which desires rapid response, easy management and security. Desktop virtualization provides a feasible solution for these problems.End users can access to virtual desktop through thin clients with less maintenance and upgrades. Moreover, user's data are stored in servers with higher redundancy level. Centralized management of virtual desktop enables faster reaction of IT operation and maintenance and facility of batch configuration. Thin clients play a vital role in user experience, which makes it important to evaluate the character of thin client. To date, several methods are developed for testing thin clients. Thin clients are designed light in weight, small in size and low in configuration, correspondingly diverse classes in software and hardware.Seven factors including hardware, remote concentrated management system, peripheral compatibility, performance, reliability, virtualization platform compatibility and energy consumption are covered in this system. VNC server should be installed in PC; clients (ARM) can access VNC servers. A VNC server uses Remote Frame buffer (RFB) protocol to send frame buffer to ARM controllers to access the PC (memories etc). Virtual Network Computing (VNC) is a graphical desktop sharing system that uses the Remote Frame Buffer protocol (RFB) to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network.

Keywords: ARM board, VNC, RDP

I. INTRODUCTION

The Remote Desktop (RD) paradigm allows end-users to remotely access content and applications running on their PCs through a network connection. When those PCs are provisioned through a virtual machine manager or hypervisor, RD turns into the Remote Virtual Desktop (RVD) paradigm. In RVD the virtual PC is rendered by a remote RVD server, and the end-user only requires a device with displaying capabilities (e.g., a physical PC, a tablet, a smartphone, etc.), a piece of software, and a network connection to access the PC desktop from any remote location. RVD services have been traditionally deployed in enterprise LAN scenarios, where network performance is optimal and the Quality of Experience (QoE) undergone by its users is generally not an issue.

With the processing of CMOS technology, the technology of video acquisition based on CMOS is becoming a new trend. However, many CMOS camera chip is not supported by the newest Linux kernel yet. In this paper, the method of designing the CMOS camera driver based on RASPBERRY developing board with the embedded Linux environment is introduced. SCCB is a distinguishing feature of OV series CMOS chips. The realization of SCCB on ARM9 embedded platform is emphasized. RASPBERRY provides a camera interface, and the camera driver is designed based on it. Then the process of the application is introduced. The experiences show that the driver works well for video previewing and Coding. The video quality can satisfy the requirements. The driver can be easily applicant on other embedded equipment.

II. REMOTE DESKTOP PROTOCOL

RDP is a proprietary protocol developed by Microsoft, which provides a user with a graphical interface to connect to another computer over a

network connection. The user employs RDP client software for this purpose, while the other computer must run RDP server software. Clients exist for most versions of Microsoft Windows (including Windows Mobile), Linux, UNIX, OS X, iOS, Android, and other modern operating systems. RDP servers are built into Windows operating systems; an RDP server for UNIX and OS X also exists. By default, the server listens on TCP port 3389 and UDP port 3389.

Remote Desktop Protocol is based on, and is an extension of, the T-120 family of protocol standards. A multichannel capable protocol allows for separate virtual channels for carrying presentation data, serial device communication, licensing information, highly encrypted data (keyboard, mouse activity), and so on. As RDP is an extension of the core T.Share protocol, several other capabilities are retained as part of the RDP, such as the architectural features necessary to support multipoint (multiparty sessions). Multipoint data delivery allows data from an application to be delivered in "real-time" to multiple parties without having to send the same data to each session individually (for example, Virtual Whiteboards). In this first release of Windows Terminal Server, however, we are concentrating on providing reliable and fast point-to-point (single-session) communications. Only one data channel will be used in the initial release of Terminal Server 4.0 However, the flexibility of RDP gives plenty of room for functionality in future products.

One reason that Microsoft decided to implement RDP for connectivity purposes within Windows NT Terminal Server is that it provides a very extensible base from which to build many more capabilities. This is because RDP provides 64,000 separate channels for data transmission. However, current transmission activities are only using a single channel (for keyboard, mouse, and presentation data). Also, RDP is designed to support many different types of Network topologies (such as ISDN, POTS, and many LAN protocols such as IPX, NetBIOS, TCP/IP, and so on). The current version of RDP will only run over TCP/IP but, with customer feedback, other protocol support may be added in future versions.

The activity involved in sending and receiving data through the RDP stack is essentially the same as the seven-layer OSI model standards for common LAN networking today. Data from an application or service to be transmitted is passed down through the protocol stacks, sectioned, directed to a channel (through MCS), encrypted, wrapped, framed, packaged onto the network protocol, and finally addressed and sent over the wire to the client. The returned data works the same way only in reverse, with the packet being stripped of its address, then unwrapped, decrypted, and so on until the data is presented to the application for use. Key portions of the protocol stack modifications occur between the fourth and seventh layers, where the data is encrypted, wrapped and framed, directed to a channel and prioritized. One of the key points for application developers is that, in using RDP, Microsoft has abstracted away the complexities of dealing with the protocol stack. This allows them to simply write clean, well-designed, well-behaved 32-bit applications, and then the RDP stack implemented by the Terminal Server and its client connections takes care of the rest.

III. PROPOSED SYSTEM

In this section, we give an overview on the proposed system architecture. Here we are using Raspberry Pi board as our platform. It has an ARM-11 SOC with integrated peripherals like USB, Ethernet and serial etc. On this board we are installing Linux operating system with necessary drivers for all peripheral devices and user level software stack which includes a light weight GUI based on XServer, XOrg middle-ware for interacting with display devices like monitors and display drivers, TCP/IP stack to communicate with network devices and some standard system libraries for system level general IO operations. The Raspberry Pi board equipped with the above software stack is connected to the outside network and a camera is connected to the Raspberry Pi through USB bus. On the other side we have to host a Server with Remote Frame Buffer facility.

- The architecture of the Server has the following layers.
- In the lower level the server has the physical Hardware.
- Above the Physical Hardware the server has Linux/Windows server software and other Remote Frame Buffer serving fecility and Control panel for user accessing.

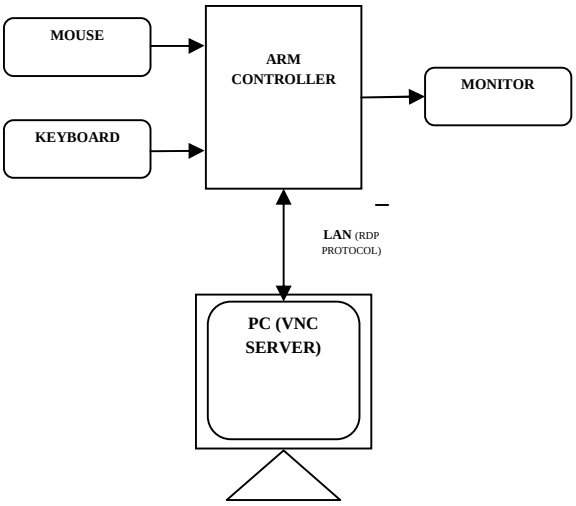


Fig.1: Proposed System

When the device starts booting from flash, it first loads the linux to the device and initializes all the drivers and the core kernel. After initialization of the kernel it first checks weather all the devices are working properly or not. After that it loads the file system and starts the startup scripts for running necessary processes and daemons. Finally it starts the main application.

When our application starts running it first check all the devices and resources which it needs are available or not. After that it checks the connection with the devices and gives control to the user. The GUI for the user has the following options. Configuration options for configuring and setting Server options which includes IP address of the server, port numbers, authentication options, session related configuration etc. these options will vary depends upon the server we have on the Internet.

- Connect Button for connecting to the Server.
- When we click the Connect button the application will check all fields are filled and configuration options are proper, then it starts connection to the server through TCP/IP protocol.

When server got the connection from the client, it first checks weather the packet is authenticated or not. If it is authenticated then it will send frame buffer to the client IP. When client got the packet from server then it will give that packet to the XORG server. The XORG server will display the frame buffer on the client screen.

IV. HARDWARE IMPLEMENTATION

A. RASPBERRY PI BOARD

The **Raspberry Pi** is a credit-card-sized single-board computer developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools.

The Raspberry Pi is manufactured in two board configurations through licensed manufacturing deals with Newark element14 (Premier Farnell), RS Components and Egoman. These companies sell the Raspberry Pi online. Egoman produces a version for distribution solely in China and Taiwan, which can be distinguished from other Pis by their red coloring and lack of FCC/CE marks. The hardware is the same across all manufacturers. The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor, Video Core IV GPU, and was originally shipped with 256 megabytes of RAM, later upgraded to 512 MB. It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and persistent storage.

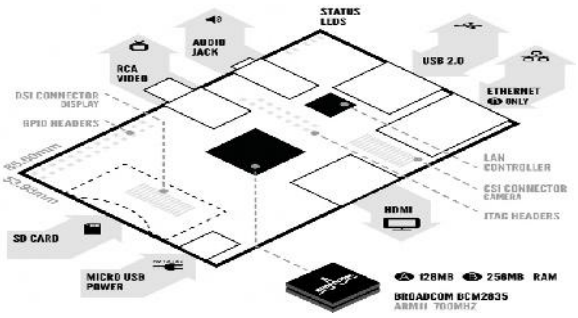


Fig2. Board features

The Foundation provides Debian and Arch Linux ARM distributions for download. Tools are available for Python as the main programming language, with support for BBC BASIC (via the RISC OS image or the Brandy Basic clone for Linux), C, Java and Perl.

B. X WINDOW SYSTEM (X SERVER)

In computing, the **X Window System** (**X11**, **X**, and sometimes informally **X-Windows**) is a windowing system for bitmap displays, common on UNIX-like operating systems. X provides the basic framework for a GUI environment: drawing and moving windows on the display device and interacting with a mouse and keyboard. X does not mandate the user interface — this is handled by individual programs. As such, the visual styling of X-based environments varies greatly; different programs may present radically different interfaces. X originated at the Massachusetts Institute of Technology (MIT) in 1984. The protocol version has been X11 since September 1987. The X.Org Foundation leads the X project, with the current reference implementation, X.Org Server, available as free and open source software under the MIT License and similar permissive licenses.

V. SOFTWARE REQUIREMENTS

A. Linux Operating System:

Linux or GNU/Linux is a free and open source software operating system for computers. The operating system is a collection of the basic instructions that tell the electronic parts of the computer what to do and how to work. Free and open source software (FOSS) means that everyone has the freedom to use it, see how it works, and changes it.There is a lot of software for Linux, and since Linux is free software it means that none of the software will put any license restrictions on users. This is one of the reasons why many people like to use Linux.

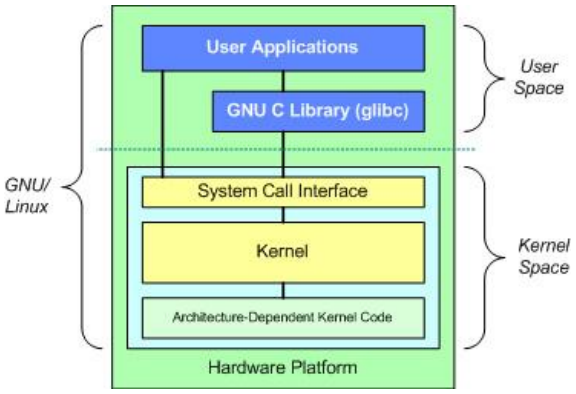


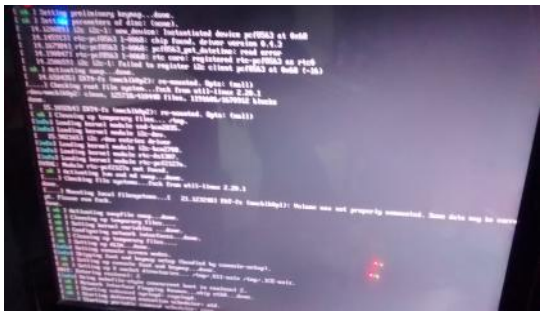
Fig7. Architecture of Linux Operating System

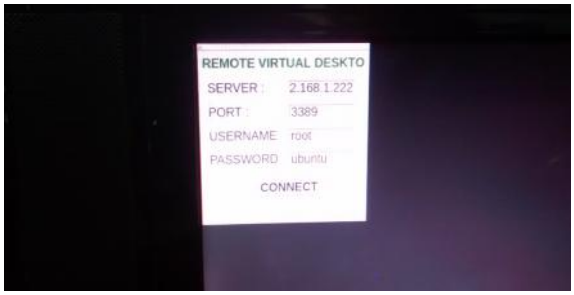
A Linux-based system is a modular Unix-like operating system. It derives much of its basic design from principles established in UNIX during the 1970s and 1980s. Such a system uses a monolithic kernel, the Linux kernel, which handles process control, networking, and peripheral and file system access. Device drivers are either integrated directly with the kernel or added as modules loaded while the system is running.

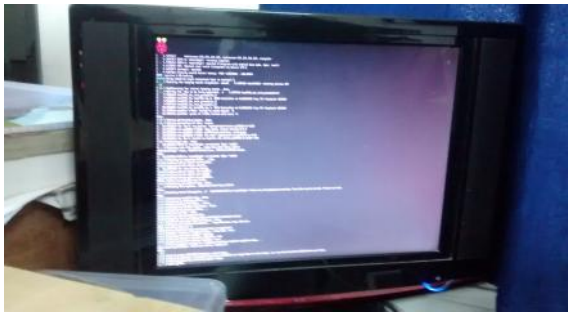
B. Qt for Embedded linux:

Qt is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI) (in which cases Qt is classified as awidget toolkit), and also used for developing non-GUI programs such ascommand-line tools and consoles for servers. Qt uses standard C++ but makes extensive use of a special code generator (called the Meta Object Compiler, or moc) together with several macros to enrich the language. Qt can also be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. Non-GUI features include SQL database access, XML parsing, thread management, network support, and a unified cross-platform application programing interface for file handling.It has extensive internationalization support.

VI. RESULTS







VII. CONCLUSION

The project "**DECENTRALIZED SOFTWARE UPGRADING IN REMOTE NETWORK BASED ON EMBEDDED LINUX**" has been successfully designed and tested. It has been developed by integrating features of all the hardware components and software used. Presence of every module has been reasoned out and placed carefully thus contributing to the best working of the unit. Secondly, using highly advanced ARM board and with the help of growing technology the project has been successfully implemented.

VIII. REFERENCES

[1] El-Gamal A., Eltoukhy H.CMOS image sensors [J].USA Circuits and Devices Magazine IEEE, 2005, 21(3): 6-20.

[2] Yuan Weiqi, Tang Yonghua. Application of SCCB of Omni Vision camera in DSP [J]. Chinese Journal of Scientific Instrument. 2006, 27 (6) 1687-1688. (In Chinese).

[3] DU Chun-lei. ARM Architecture and Programming [M].Beijing: Tsinghua University Press, 2003. (In Chinese).

[4] Zhang Da-bo. Principles, Design and Application of Embedded System [M].Beijing: Machinery Industry Press, 2004. (In Chinese).

[5] Samsung. S3C2440A 32Bit CMOS Microcontroller User's Manual [Z]. Samsung Electronics Corp, 2003.

[6] Jonathan C, Alessandro R, Greg K. Linux device drivers [M].3rd ed. Sebastopol, CA:O' Reilly & Associates, 2006:324-327.

[7] Daniel P, Cesati M. Understanding the Linux kernel [M]. 3rd ed. Sebastopol, CA: O' Reilly & Associates, 2006:525.

[8] Omni Vision Serial Camera Control Bus (SCCB) Functional Specification. Omni Vision Technologies. Version: 2.1, 2003, 2.

M.Srikanth received the B.Tech degree in Electronics &Communication Engineering from LIET, JNTU Hyderabad in 2010, and pursuing M.Tech in Embedded Systems from MRTN, JNTU, and Hyderabad.

.

M.Vijaya Lakshmi received the B.Tech degree in Electronics & Communication Engineering from GPREC, Kurnool in 2000, and Masters in Electronics & Communication Engineering from JNTU, Hyderabad.